Bare Metal C Programming

Young Won Lim
1/23/15

Please send corrections (or suggestions) to youngwlim@hotmail.com.

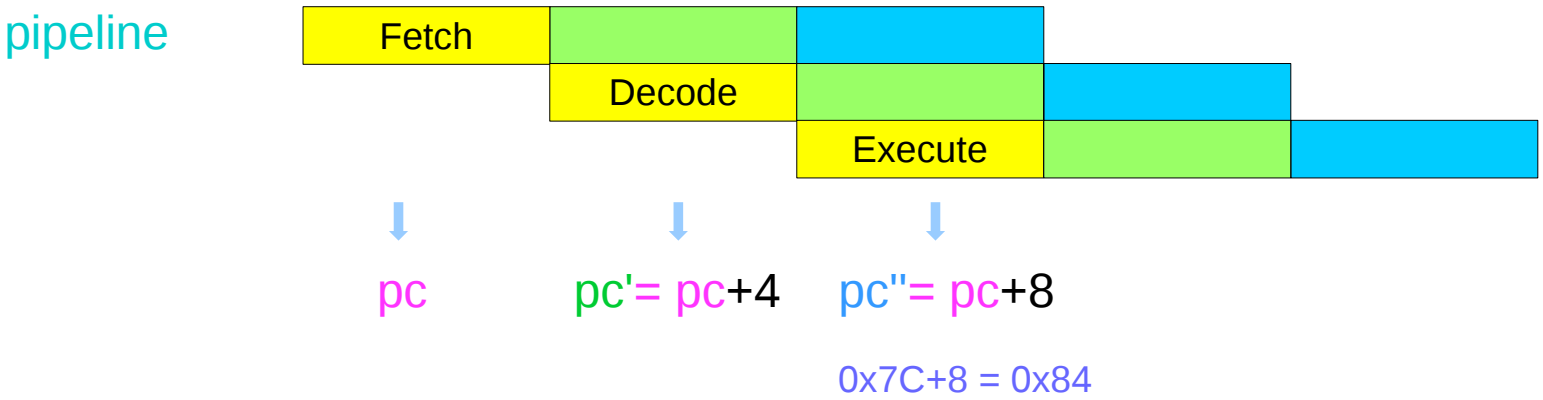
This document was produced by using OpenOffice.

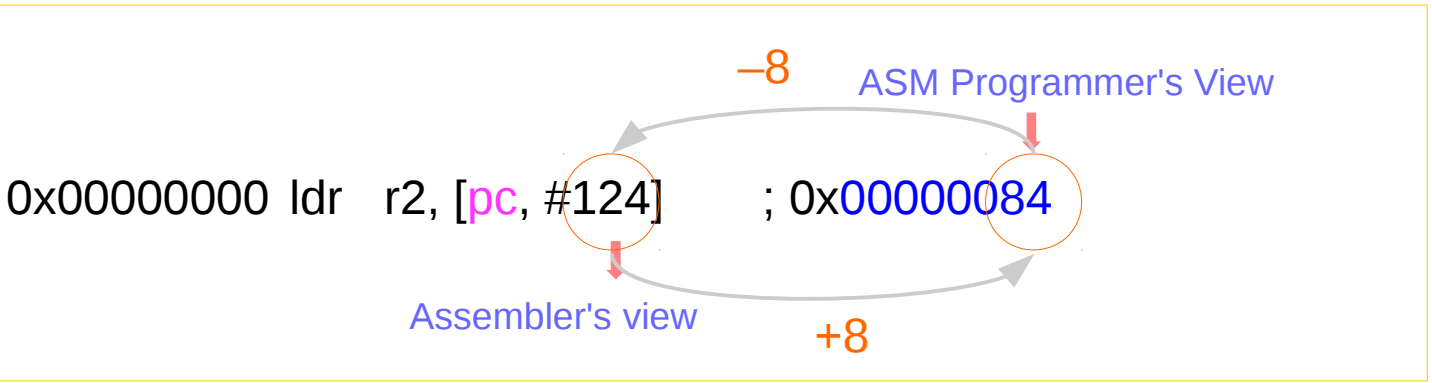
PC Relative Addressing

```
0x00000000  ldr   r2, [pc, #124]      ; 0x00000084
```

124_{10} = 0x7C

132_{10} = 0x84





http://www.valvers.com/open-software/raspberry-pi/step02-bare-metal-programming-in-c-pt2/

Memory Alignment (1)

```
0x00000000  ldr   r2, [pc, #124]      ; 0x00000084
0x00000004  ldr   r3, [pc, #124]      ; 0x00000088
0x00000008  ldr   r0, [r2]

0x00000084  00010090
0x00000088  00010094

0x00008090  20200000
```

executable

bloated code

```
0x00008000  ldr   r2, [pc, #124]      ; 0x00000084
0x00008004  ldr   r3, [pc, #124]      ; 0x00000088
0x00008008  ldr   r0, [r2]

0x00008084  00010090
0x00008088  00010094

0x00010090  20200000 ---> actual address 0x10090
```

memory

http://www.valvers.com/open-software/raspberry-pi/step02-bare-metal-programming-in-c-pt2/

Memory Alignment (2)

```
0x00000000  ldr   r2, [pc, #128]; 0x00000088
0x00000004  ldr   r3, [pc, #128]; 0x0000008c
0x00000008  ldr   r0, [r2]


0x00000088  00008094
0x0000008C  00008098

0x00000094  20200000
```

executable

compact code

```
0x00008000  ldr   r2, [pc, #128]; 0x00000088
0x00008004  ldr   r3, [pc, #128]; 0x0000008c
0x00008008  ldr   r0, [r2]


0x00008088  00008094
0x0000808C  00008098

0x00008094  20200000
```

memory

http://www.valvers.com/open-software/raspberry-pi/step02-bare-metal-programming-in-c-pt2/

Memory Alignment (3)

Linker Script

```
/* Adjust the address for the data segment.
   We want to adjust up to
   the same address within the page on the next page up.  */

. = ALIGN( CONSTANT (MAXPAGESIZE) ) + (. & (CONSTANT (MAXPAGESIZE) – 1) );
```

ALIGN doesn't change the value of the location counter--it just does arithmetic on it.
As an example, to align the output .data section
to the next 0x2000 byte boundary after the preceding section and
to set a variable within the section to the next 0x8000 boundary after the input sections:

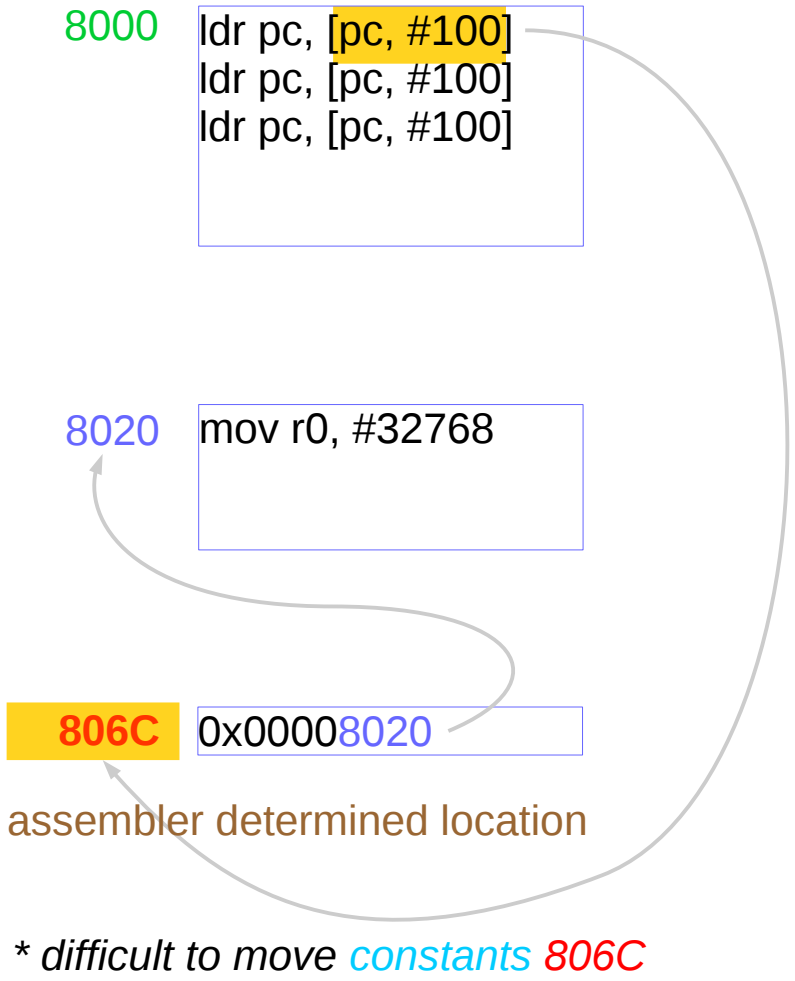
```
SECTIONS{ ...
  .data ALIGN(0x2000): {
    *(.data)
    variable = ALIGN(0x8000);
  }
... }
```

The first use of ALIGN in this example specifies the location of a section because it is used as the optional start attribute of a section definition.
The second use simply defines the value of a variable.
The built-in NEXT is closely related to ALIGN.

Indirect Addressing – PC Relative

```
_start:
        ldr    pc,    =_reset_

_reset_:
        ldr    r0,    =_start
        ldr    r1,    #0x0000
        ldmia        r0!, {r2, r3, r4, r5, r6, r7, r8, r9}
        stmia        r1!, {r2, r3, r4, r5, r6, r7, r8, r9}
```



** difficult to move constants 806C*

Moving Vector Tables – Assembly

```
_start:
    ldr    pc,    =_reset_
    ldr    pc,    =undef_vector
    ldr    pc,    =swi_vector
    ldr    pc,    =prefetch_abort
    ldr    pc,    =_reset_
    ldr    pc,    =_reset_
    ldr    pc,    =int_vector
    ldr    pc,    =fiq_vector
```

```
_reset_:
    ldr    r0,    =_start
    ldr    r1,    #0x0000
    ldmia        r0!, {r2, r3, r4, r5, r6, r7, r8, r9}
    stmia        r1!, {r2, r3, r4, r5, r6, r7, r8, r9}

    * constants 806C has not be moved!
```

```
00008000 <_start>:
    8000:      ldr pc, [pc, #100]    ; 806c <_enable_interrupts +0x10>  806C    = 805C + 10
    8004:      ldr pc, [pc, #100]    ; 8070 <_enable_interrupts +0x14>  8070    = 805C + 14
    8008:      ldr pc, [pc, #100]    ; 8074 <_enable_interrupts +0x18>  8074    = 805C + 18
    800C:      ldr pc, [pc, #100]    ; 8078 <_enable_interrupts +0x1c>  8078    = 805C + 1C
    8010:      ldr pc, [pc, #84]     ; 806c <_enable_interrupts +0x10>  806C    = 805C + 10
    8014:      ldr pc, [pc, #80]     ; 806c <_enable_interrupts +0x10>  806C    = 805C + 10
    8018:      ldr pc, [pc, #92]     ; 807c <_enable_interrupts +0x20>  807C    = 805C + 20
    801C:      ldr pc, [pc, #92]     ; 8080 <_enable_interrupts +0x24>  8080    = 805C + 20

00008020 <_reset_>:
    8020:      mov r0, #32768        ; 0x8000
```

Moving Vector Tables – Assembled Code

```
00008000 <_start>:
    8000:    ldr pc, [pc, #100]    ; 806c <_enable_interrupts +0x10>  [_reset_]
    8004:    ldr pc, [pc, #100]    ; 8070 <_enable_interrupts +0x14>  [undef_vector]
    8008:    ldr pc, [pc, #100]    ; 8074 <_enable_interrupts +0x18>  [swi_vector]
    800C:    ldr pc, [pc, #100]    ; 8078 <_enable_interrupts +0x1c>  [prefetch_abort]
    8010:    ldr pc, [pc, #84]     ; 806c <_enable_interrupts +0x10>  [_reset_]
    8014:    ldr pc, [pc, #80]     ; 806c <_enable_interrupts +0x10>  [_reset_]
    8018:    ldr pc, [pc, #92]     ; 807c <_enable_interrupts +0x20>  [int_vector]
    801C:    ldr pc, [pc, #92]     ; 8080 <_enable_interrupts +0x24>  [fiq_vector]

00008020 <_reset_>:
    8020:    mov r0, #32768        ; 0x8000
```

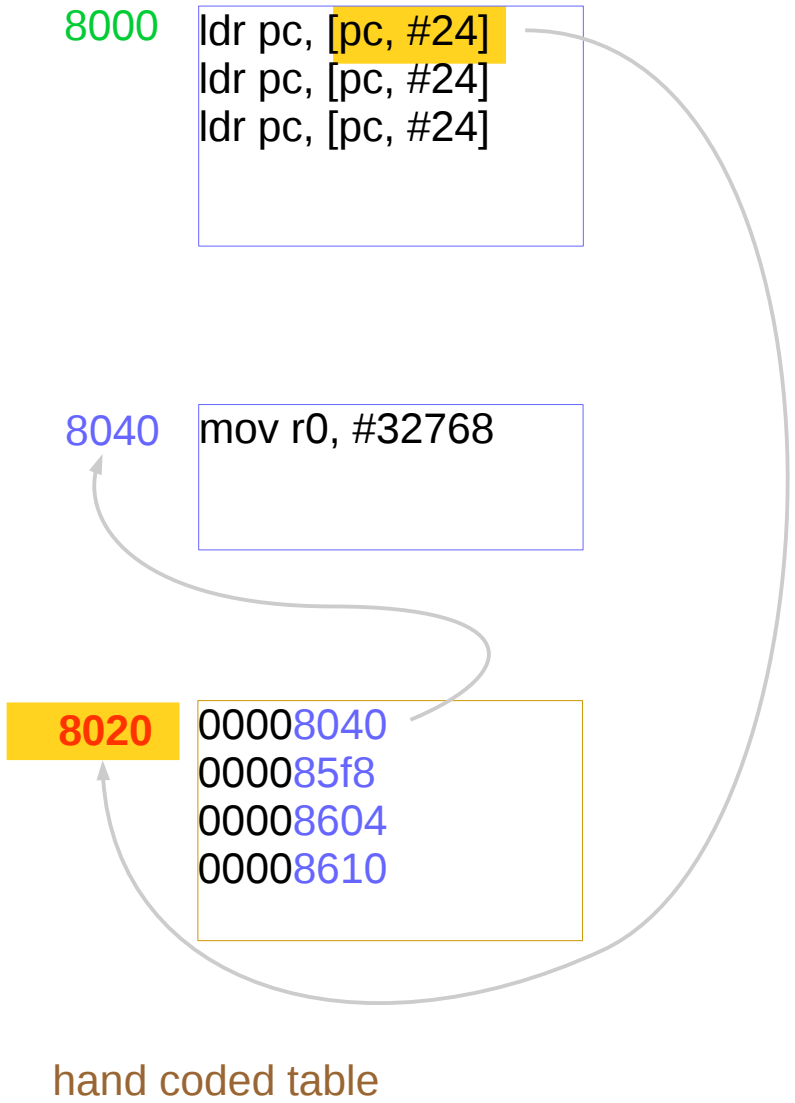
```
0000805c <_enable_interrupts>:
    805C:    mrs  r0,         CPSR
    8060:    bic  r0,         r0, #128   ;   0x80
    8064:    msr  CPSR_c, r0
    8068:    mov pc,          lr
    806C:    .word   0x00008020      ;   8020    =_reset_
    8070:    .word   0x000085e8      ;   85e8    =undef_vector
    8074:    .word   0x000085f4      ;   85f4    =swi_vector
    8078:    .word   0x00008600      ;   8600    =prefetch_abort
    807C:    .word   0x00008628      ;   8628    =int_vector
    8080:    .word   0x0000869c      ;   869c    =fiq_vector
```

Indirect Addressing – using .word table

```
_start:
        ldr     pc,     =_reset_h
        ldr     pc,     =_undef_vector_h
        ldr     pc,     =_swi_vector_h
        ldr     pc,     =_prefetch_abort_h
```

```
    8020                            8040
_reset_h:               .word   _reset_
_undef_vector_h:        .word   undef_vector
_swi_vector_h:          .word   swi_vector
_prefetch_abort_h:      .word   prefetch_abort
```

```
_reset_:
        ldr     r0,     =_start
        ldr     r1,     #0x0000
        ldmia           r0!, {r2, r3, r4, r5, r6, r7, r8, r9}
        stmia           r1!, {r2, r3, r4, r5, r6, r7, r8, r9}
```



hand coded table

Coding Vector Tables – Assembly

```
_start:
        ldr     pc,     =_reset_h
        ldr     pc,     =_undef_vector_h
        ldr     pc,     =_swi_vector_h
        ldr     pc,     =_prefetch_abort_h
        ldr     pc,     =_data_abort_h
        ldr     pc,     =_unused_h
        ldr     pc,     =_int_vector_h
        ldr     pc,     =_fiq_vector_h
```

```
_reset_h:           .word   _reset_
_undef_vector_h:    .word   undef_vector
_swi_vector_h:      .word   swi_vector
_prefetch_abort_h:  .word   prefetch_abort
_data_abort_h:      .word   data_abort
_reset_h:           .word   _reset_
_int_vector_h:      .word   int_vector
_fiq_vector_h:      .word   fiq_vector
```

```
_reset_:
        ldr         r0,     #0x8000
        ldr         r1,     #0x0000
        ldmia       r0!,    {r2, r3, r4, r5, r6, r7, r8, r9}      ; ldr pc, =xxx
        stmia       r1!,    {r2, r3, r4, r5, r6, r7, r8, r9}
        ldmia       r0!,    {r2, r3, r4, r5, r6, r7, r8, r9}      ; xxx .word yyy
        stmia       r1!,    {r2, r3, r4, r5, r6, r7, r8, r9}
```

Coding Vector Tables – Assembled Code

```
00008000 <_start>:
    8000:      ldr pc, [pc, #24]      ; 8020 <_reset_h>
    8004:      ldr pc, [pc, #24]      ; 8024 <_undef_vector_h>
    8008:      ldr pc, [pc, #24]      ; 8028 <_swi_vector_h>
    800C:      ldr pc, [pc, #24]      ; 802c <_prefetch_abort_h>
    8010:      ldr pc, [pc, #24]      ; 8030 <_data_abort_h>
    8014:      ldr pc, [pc, #24]      ; 8034 <_unused_h>
    8018:      ldr pc, [pc, #24]      ; 8038 <int_vector_h>
    801C:      ldr pc, [pc, #24]      ; 803c <fiq_vector_h>
```

```
00008020 <_reset_h>:
    8020:  00008040    .word   0x00008040

00008024 <_undef_vector_h>:
    8024:  000085f8    .word   0x000085f8

00008028 <_swi_vector_h>:
    8028:  00008604    .word   0x00008604

0000802c <_prefetch_abort_h>:
    802c:  00008610    .word   0x00008610
```

```
00008030 <_data_abort_h>:
    8030:  00008624    .word   0x00008624

00008034 <_unused_h>:
    8034:  00008040    .word   0x00008040

00008038 <int_vector_h>:
    8038:  00008638    .word   0x00008638

0000803c <fiq_vector_h>:
    803c:  000086ac    .word   0x000086ac
```

```
00008040 <_reset_>:
    8040:  e3a00902    mov r0, #32768  ;
```

Address Type Casting (1)aaaaaaaaa

References

[1] http://wiki.osdev.org/ARM_RaspberryPi_Tutorial_C
[2] http://blog.bobuhiro11.net/2014/01-13-baremetal.html
[3] http://www.valvers.com/open-software/raspberry-pi/
[4]